WEST

Generate Collection | Print

L3: Entry 11 of 17 File: USPT Dec 5, 2000

US-PAT-NO: 6157996
DOCUMENT-IDENTIFIER: US 6157996 A

TITLE: Processor programably configurable to execute enhanced variable byte length instructions including predicated execution, three operand addressing, and increased register space

DATE-ISSUED: December 5, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Christie; David S.	Austin	TX		
Kranich; Uwe	Munich			DE

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Advanced Micro Devices, Inc.	Sunnyvale	CA			02

APPL-NO: 08/ 969779 [PALM]
DATE FILED: November 13, 1997

INT-CL: [07] G06 F 9/38

US-CL-ISSUED: 712/218; 712/219, 712/215, 712/208, 712/234
US-CL-CURRENT: 712/218; 712/208, 712/215, 712/219, 712/234

FIELD-OF-SEARCH: 712/32, 712/24, 712/36, 712/1, 712/214, 712/215, 712/204, 712/208, 712/209, 712/210, 712/219, 712/234

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected | Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4044338	August 1977	Wolf	
☐	4050094	September 1977	Bourke	
☐	4385352	May 1983	Bienvenu	
☐	4453212	June 1984	Gaither et al.	
☐	4807115	February 1989	Torng	
☐	4835734	May 1989	Kodaira	
☐	4858105	August 1989	Kuriyama et al.	
☐	4926322	May 1990	Stimac	
☐	4928223	May 1990	Dao et al.	
☐	4972338	November 1990	Crawford	
☐	5053631	October 1991	Perlman et al.	
☐	5058048	October 1991	Gupta et al.	
☐	5109334	April 1992	Kamuro	
☐	5125087	June 1992	Randell	
☐	5129067	July 1992	Johnson	
☐	5136697	August 1992	Johnson	
☐	5226126	July 1993	McFarland et al.	
☐	5226130	July 1993	Favor et al.	
☐	5226132	July 1993	Yamamoto	
☐	5274834	December 1993	Kardach	
☐	5293592	March 1994	Fu et al.	
☐	5321836	June 1994	Crawford	
☐	5375213	December 1994	Arai	
☐	5438668	August 1995	Coon et al.	
☐	5471593	November 1995	Branigin	395/375
☐	5481684	January 1996	Richter	
☐	5560032	September 1996	Nguyen	
☐	5561784	October 1996	Chen	
☐	5651125	July 1997	Witt et al.	
☐	5758116	May 1998	Lee et al.	712/210
☐	5809273	September 1998	Favor et al.	712/210
☐	5838984	November 1998	Nguyen et al.	712/5
☐	5848284	December 1998	Sharangpani	712/1

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
0 67667	December 1982	EP	
0259095	March 1988	EP	
0 369396	May 1990	EP	
0381471	August 1990	EP	
0 425410	May 1991	EP	
0459232	December 1991	EP	
0 467152	January 1992	EP	
2263987	August 1993	GB	
2263985	August 1993	GB	
2281422	March 1995	GB	

OTHER PUBLICATIONS

Intel 486 Dx Microprocessor (referred as Intel) Analysis Techniques for Predicated Code.
Intel, "Chapter 2: Microprocessor Architecture Overview," 1994, pp. 2-1 through 2-4.
Michael Slater, "AMD's K5 Designed to Outrun Pentium," Microprocessor Report, vol. 8, No. 14, Oct. 24, 1994, 7 pages.
Sebastian Rupley and John Clyman, "P6: The Next Step?," PC Magazine, Sep. 12, 1995, 16 pages.
Tom R. Halfhill, "AMD K6 Takes On Intel P6," BYTE, Jan. 1996, 4 pages.
"Intel Architecture Software Developer's Manual, Volume 1: Basic Architecture", Intel Corporation, Prospect IL, 1996, 1997, Chapter 8: Programming With The Intel MMX.TM. Technology, pp. 8-1 through 8-15.
Holstad, S., "Tutorial Tuesday: Decoding MMX" Jan. 14, 1997, Earthlink Network, Inc. copyright 1997, 5 pages (see http://www.earthlink.net/daily/Tuesday/MMX).
"Intel MMX.TM. Technology--Frequently Asked Questions" 6 pages (see http://www.intel.com/drg/mmx/support/faq/htm).
Kathail et al., HPL Playdoh Architecture Specification: Version 1.0, Hewlett Packard, Computer Systems Laboratory, HPL-93-80, Feb., 1994, pp. 1-48.
Intel Microprocessors: vol. I, 1993, pp. 2-1, 2-18 to 2-26, 2-79 to 2-83 and 2-121 to 2-122.
Intel Microprocessors: vol. II, 1993, pp. 2-2, 2-19 to 2-26, 2-80 to 2-83 and 2-122.
IEEE Micro, vol. 13, No. 5, Oct. 1, 1993, pp. 24-36, Makato Awaga et al, "The VP 64-Bit Vector Coprocessor: A New Implementation of High-Performance Numerical Computation."

ART-UNIT: 273

PRIMARY-EXAMINER: An; Meng-Ai T.

ASSISTANT-EXAMINER: Monestime; Mackly

ATTY-AGENT-FIRM: Conley, Rose & Tayon Merkel; Lawrence J.

ABSTRACT:

A processor for executing computer instructions including, in one embodiment, a machine specific register (MSR) which includes a predicated execution field and an instruction decoder. The decoder is coupled to the MSR and configured to detect predicated execution information contained in the computer instruction and to include conditional execution information in the decoded instruction upon detecting an appropriate setting in the predicated execution field of the MSR. The processor further includes a first execution unit. The first execution unit is configured to detect and evaluate the conditional execution information in the decoded instruction and, if present, to execute the decoded instruction only if a condition represented by the conditional execution information is true. In another embodiment, the processor includes a standard register set and an extended register set, which includes the standard register set. The decoder is configured to search the computer instruction for an extended register indicator upon detecting an appropriate setting in the extended register field of the MSR. The decoder is further configured to fetch, upon

detecting the extended register indicator, a value from a selected register within the extended register set. If the decoder detects the absence of extended register indicator, a value is fetched from a selected register where the selected register is within the standard register set. In another embodiment, the MSR includes a three register field and the decoder is configured to interpret the computer instruction as containing first and second source register operands and a destination operand if the instruction contains a three register indicator and the three register field is set appropriately.

22 Claims, 14 Drawing figures

L3: Entry 11 of 17 File: USPT Dec 5, 2000

DOCUMENT-IDENTIFIER: US 6157996 A
TITLE: Processor programably configurable to execute enhanced variable byte length instructions including predicated execution, three operand addressing, and increased register space

Assignee Name (1):
Advanced Micro Devices, Inc.

Brief Summary Text (10):
Another limitation of the existing .times.86 instruction is the inability to predicate execution of instructions. Predicated execution refers to a situation in which an instruction is executed if and only if a predicate condition is met, wherein the condition to be evaluated is part of the instruction itself. Predicated execution of instructions can increase performance of highly pipelined microprocessor architectures by minimizing branch misprediction and its attendant performance penalties. In a pipelined microprocessor architecture, the microprocessor is preparing and executing multiple instructions in each clock cycle. As an example, a simplistic microprocessor pipeline might include four stages: fetch, decode, execute, and writeback. During any given clock cycle, the microprocessor is fetching a first instruction from an instruction cache, decoding a second instruction, executing a third instruction, and writing back the results of a previously executed fourth instruction to a register file or a cache memory. To keep the pipeline filled, the processor must determine which instructions are most likely to be executed following the instruction that is currently executing. This determination is less than precise because computer programs typically do not execute instructions in a linear or otherwise predictable manner. Instead, a typical computer program includes at least one decision step in which the result of the decision step determines which instruction will execute next. Prior to the actual execution of such a decision step, the microprocessor must attempt to predict which step will be executed after the decision step. When the processor mispredicts (i.e., when the instruction predicted by the processor to be executed after the decision step turns out not to be the correct instruction), a performance penalty is paid in a pipelined processor because the pipeline must be cleared resulting in the occurrence of one or more no-op cycles. A no-op cycle, for purposes of this disclosure refers to a processor clock cycle during which no instruction is executed by the processor. As an example using the four stage pipeline proposed earlier, it is possible that the condition represented by a decision step is not fully evaluated until the fourth or writeback stage. In such an embodiment, misprediction requires that the instructions in the previous three stages of the pipeline be cleared. The performance penalty for misprediction increases as the number of stages in the pipeline increases. Accordingly, it is desirable to minimize the occurrence of misprediction in a pipelined processor without placing any significant restrictions on the ability of systems and applications programmers to insert decision steps in their code.

Brief Summary Text (14):
The problems identified above are in large part addressed by a microprocessor for executing computer instruction where the microprocessor includes, in various embodiments, facilities for increasing the addressable register space, conditionally executing instructions based upon predicate information contained in the computer instruction, and a facility for operating in a three register operand mode in which the instruction operands include a first and second source operand as well as a third destination operand.

Brief Summary Text (15):
Broadly speaking, the present invention contemplates a processor for executing a computer instruction. The processor includes a machine specific register (MSR). The MSR includes a predicated execution field. The processor further includes an instruction decoder adapted to receive the computer instructions and to produce a decoded instruction upon receiving the computer instruction. The decoder is coupled to the MSR and the decoder is configured to search the computer instruction for predicated execution information upon detecting an appropriate setting in the predicated execution field. The decoder is further configured to include, upon detecting the predicated information in the computer instruction, conditional execution information in the decoded instruction. A first execution unit of the processor is configured to receive and execute the decoded instruction from the instruction decoder. The first execution unit is coupled to the MSR and configured to detect and evaluate the conditional execution information in the decoded instruction in response to detecting the appropriate setting in the predicated execution field of the MSR. The first execution unit is configured to execute the decoded instruction only if a condition represented by the conditional execution information is true. In one embodiment, a byte length of the computer's instruction is variable and the instruction decoder is configured to determine a beginning byte and an ending byte of the variable length computer instruction. In one embodiment, the predicated execution information is contained within a beginning byte of the computer instruction. In the presently preferred embodiment, the predicated execution field of the MSR consists of a single conditional execution bit wherein the execution information is included in the decoded instruction if the conditional execution grid is set. In one embodiment, the processor further includes at least one predicate FLAG. In this embodiment, the first execution unit is configured to set or clear the predicate FLAG to reflect a characteristic of a result produced by the execution of the decoded instruction if a predicate FLAG field within the MSR is set appropriately. Preferably, the predicate FLAG field consists of a predicate FLAG bit where the status of the predicate FLAG bit determines whether the execution unit sets or clears the predicate FLAG upon executing the decoded instruction. In one embodiment, the truth of the condition depends upon the state of the predicate FLAG prior to the execution of the computer instruction. In other words, the truth of the condition depends upon the state of the predicate FLAG which was set during the execution of a previous computer instruction.

Detailed Description Text (6):
In the embodiment shown in FIG. 5, processor 410 includes an EFLAG register 520 and a predicate FLAG register 522. An EFLAG register will be familiar to those familiar with .times.86 type processor architectures and contains a set of readable FLAGs that are set after the execution of an appropriate computer instruction such that the individual FLAGs within EFLAG register 520 represent various characteristics of a result produced by the execution of the computer instruction. For example, if the execution of a particular instruction produces a result that is equal to zero, a zero FLAG within EFLAG 510 is set accordingly. A subsequent computer instruction may the read the state of a particular FLAG to determine, in a rapid fashion, certain characteristics of the preceding operation. The individual FLAGs within a typical .times.86 type EFLAG register are particularly useful in computer instructions such as jump instructions which modify the instruction pointer (not shown in the drawings) of CPU 410. The embodiment of CPU 410 shown in FIG. 5 includes a substantial duplicate of EFLAG's register 520 identified as predicate FLAG (PFLAG) register 522. PFLAG register 522, as discussed in greater detail below, may be useful in reducing branch misprediction and therefore may be useful in improving the performance of a pipeline processor such as processor 410. Processor 410 further includes a machine specific register (MSR). An MSR is a register that is not defined in a conventional .times.86 architecture. Examples of registers found in a conventional .times.86 architecture can be found in numerous industry publications including, for example, the registers identified in Crawford and Gelsinger, Programming the 80386 (Sybex 1987). The inclusion of MSR 524 in processor 410 facilitates the implementation of an enhanced instruction set useful for extending capabilities and eliminating the limitations of a conventional .times.86 processor. To maximize the marketability of processor such as processor 410 which include machine specific registers, such processors typically are designed to be compatible with existing software written for industry standard .times.86 microprocessors. It is contemplated that declaring the specific fields described below within MSR 524, processor 410 is capable of executing software code written for conventional .times.86 architectures. Furthermore, while some of the

specific embodiments described in the following discussion imply an .times.86 type instruction format, it is to be understood that, in the broadest sense, the following processor enhancements are generally applicable to a broader class of processors. The presence of MSR 524 within processor 410 implies, at a minimum, an instruction set extension in which the specific fields and bits within MSR 524 are read and written.

Detailed Description Text (14):
MSR 524 further includes a predication update field which is routed to instruction execution logic 618. If predicated update field 626 is set appropriately, instruction execution logic 618 will set not only EFLAG register 620 appropriately, but also PFLAG register 622 depending upon the result generated by the execution of decoded instruction 608. This PFLAG value may then be used by a subsequent predicated instruction for conditional execution purposes. In the embodiment shown in FIG. 6, execution 610 is coupled to a write back unit designed to form the actual updating of the EFLAG and PFLAG registers if the update field 626 of MSR 524 contains an appropriate value. In a presently preferred embodiment, both the conditional execution field 602 and the predicated update field 626 of MSR 524 comprise a 1 bit field. The status of the predicate FLAG bit, in this embodiment, determines whether the execution unit sets or clears predicate flag 622 upon executing decoded instruction 608.

Detailed Description Text (15):
Those familiar with .times.86 instruction sets will recognize that the format 710 of an instruction suitable for use with processor 410 as shown in FIG. 6 closely resembles the instruction format of an .times.86 instruction as shown in FIG. 1. It is contemplated, therefore, that in a presently preferred embodiment, computer instruction 606 is a variable byte length instruction and it is further contemplated that instruction decoder 604 includes an instruction decode unit 612 that is capable of determining a beginning byte and an ending byte of computer instruction 606. The seven conditions labeled as CC1-CC7, in a preferred embodiment, represent the status of either the EFLAGs within EFLAGs register 620 or the PFLAGs within PFLAGs register 622. In the preferred embodiment, processor 410 updates the EFLAG register 620 whenever the PFLAG register 622 are updated. In another embodiment, however, it is contemplated that the PFLAGs may be set independently of the EFLAGs within EFLAGs register 620. In the preferred embodiment, the truth of each of the condition codes CC1 through CC7 depends upon the state of a predicate FLAG 622. In one embodiment, for example, the prefix bytes 40H through 4fH are assigned as the predication execution prefixes. If an instruction 606 includes a prefix byte of 41H, the CC1 condition is checked prior to the execution of decoded instruction 608 and PFLAGs register 622 is not updated by the execution of decoded instruction 608. The CC1 condition may represent; as an example, whether the zero flag (ZF) in PFLAGs register 622 is set. Accordingly, prior to execution decoded instructions 608, instruction execution logic 618 extracts the conditional execution information 624 and routes this information to conditional information evaluator 616. Conditional information evaluator 616, which is coupled to PFLAGs register 622 performs a simple comparison with the state of ZF in PFLAGs register 622 and returns a result of this comparison to instruction execution logic 618. If the result produced by conditional information evaluator 616 indicates that the condition checked was true, i.e., indicates that ZF of PFLAGs register 622 was set, instruction execution logic 618 executes decoded instruction 608. Referring to Table 712 of FIG. 7, it is seen, in this embodiment, that up to 8 different conditions may be checked or monitored. It is further seen from the format indicated by the format of instruction 710, that this embodiment contemplates an instruction format essentially equivalent to the instruction format of a standard .times.86 instruction as diagrammed in FIG. 1 thereby minimizing the alterations and extensions required to be made in instruction decode unit 612 and instruction execution logic 618.

Detailed Description Text (23):
Where the .sub.-- U suffix indicates that the predicate flags are updated and the CC#.sub.-- prefix indicates conditional execution of the associated instruction depending upon the truth of the condition represented by CC#. It should be apparent that the predicated sequence guarantees linear program flow. In other words, in the predicated sequence, I2.1 is guaranteed to be the instruction that will be conditionally executed after execution of I2, which, in turn, is the instruction guaranteed to be conditionally executed after I1. By eliminating branch misprediction, therefore, predicate execution improves system performance. It is also noteworthy that

the addition of a predicate flag register insures that the instruction I11 may result in a change to the regular EFLAGs without affecting the conditional execution of instruction I12 which remains dependent solely on the comparison made in I1.

Detailed Description Text (27):
Turning briefly to FIG. 13, a diagram for an embodiment of the present invention incorporating the predicated execution enhancements and the extended register enhancements previously described into a single processor are implemented by the modification of the extended register byte described with respect to FIG. 9. Predicated execution and extended register addressing can be incorporated into a single processor 410 by combining the features of decoder 604 (FIG. 6) and decoder 802 (FIG. 8). In an embodiment of processor 410 designed to be used with the instruction format diagrammed in FIG. 13, the presence of extended register and predicated execution information as indicated by a prefix byte containing a hexadecimal value 40. Other methods of indicating the presence of the extended register information and the predicated execution can be easily accommodated. For example, the prefix byte used to indicate instruction set enhancements may be a different value than hexadecimal 40. In addition, the instruction enhancements may be indicated in any location within the instruction other than the prefix byte. The prefix byte is used in the embodiment shown to simplify the logic required to extract the instruction enhancement indicators. The format of FIG. 13 includes an instruction enhancement byte (IEB byte) which includes information to implement both the extended registers and the conditional execution described previously. Referring specifically to the diagram of the IEB byte shown in FIG. 13, and the underlying table, it is seen that the high order three bits of the IEB byte include the conditional execution information described with respect to FIG. 6. The next most significant bit of IEB byte contains the bit dedicated to indicating whether the predicate flag field will be updated by an executed instruction. The low order four bits include the extended register information including the upper two bits of the R/M register field and the REG register field as described previously with respect to FIG. 8.

CLAIMS:

5. The processor of claim 1, wherein said processor includes at least one predicate flag, and wherein said first execution unit is configured to set or clear said predicate flag to reflect a characteristic of a result produced by said execution of said decoded instruction if a predicate flag field within said MSR is set appropriately.

6. The processor of claim 5, wherein said predicate flag field consists of a predicate flag bit, wherein the status of said predicate flag bit determines whether said execution unit sets or clears said predicate flag upon executing said decoded instruction.

7. The processor of claim 6, wherein the truth of said condition depends upon the state of said predicate flag, wherein said state of said predicate flag was set during the execution of a previous instruction.